DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

MULTIPLE SENSOR TRACKING IN THE
INTERIM BATTLE GROUP TACTICAL TRAINER

by

Keith N. Spangenberg

March 1985

Thesis Advisor: Rex H. Shudde

Approved for public release; distribution is unlimited

T223270

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Multiple Sensor Tracking in the Interim Battle Group Tactical Trainer		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1985 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Keith N. Spangenberg		8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE March 1985 13. NUMBER OF PAGES 61
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Kalman Filter, Tracking Model, Interim Battle Group Tactical Trainer, Naval Warfare Interactive Simulation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The thesis provides two subroutines for the Interim Battle Group Tactical Trainer (IBGTT). The first subroutine is a single sensor tracking model using the Kalman filter. This subroutine is part of the Passive Sonar Model. The second subroutine is a multiple sensor tracking model using the Kalman filter to correlate all sensor contacts on a specific target. This subroutine is a separate entity and can be turned

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

 SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. Continued

on or off at simulation initiation as required by training objectives.

Approved for public release; distribution is unlimited.

Multiple Sensor Tracking in the Interim Battle Group Tactical Trainer

by

Keith N. Spangenberg
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEM TECHNOLOGY
(ANTISUBMARINE WARFARE)

from the

NAVAL POSTGRADUATE SCHOOL
March 1985

ABSTRACT

The thesis provides two subroutines for the Interim Battle Group Tactical Trainer (IBGTT). The first subroutine is a single sensor tracking model using the Kalman filter. This subroutine is part of the Passive Sonar Model. The second subroutine is a multiple sensor tracking model using the Kalman filter to correlate all sonar contacts on a specific target. This subroutine is a separate entity and can be turned on or off at simulation initiation as required by training objectives.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

The Naval Ocean Systems Center has developed the Interim Battle Group Tactical Trainer/Computer Support Facility (IBGTT/CSF) as a computer-based tactical simulation system to provide a training device for senior naval officers to practice tactical decision making until such time as the Enhanced Naval Warfare Gaming System becomes available. The trainer is intended to provide interactive, multithreat, multiplatform operational situations in a simulated yet realistic operational environment so that selected officers can study, practice, and be evaluated in force-level tactical decision making.

The IBGTT training capability is implemented as a real-time, man-interactive, computer-aided (discrete event, time step) simulation of the naval warfare environment. In operation, the IBGTT supports a two-sided (BLUE vs. ORANGE) interactive scenario in which opposing sides can define, structure, and dynamically control forces ranging in size from one or more battle groups and associated aircraft, down to a single air or surface unit. Force elements and their associated characteristics, sensors, weapons, and communication systems may be derived from real, proposed, or conceptualized units or systems.

The utilization of IBGTT involves the use of the four major Naval Warfare Interactive Simulation System (NWISS) processes; BUILD, FORCE, WARGAME, and POST-GAME ANALYSIS. The BUILD process is a stand alone interactive program used to create and maintain platform, sensor, communication, and weapon characteristics in the IBGTT Characteristic Data Base. The FORCE process is a stand alone interactive program used to create and maintain an exercise scenario using the

data base created by the BUILD process. The WARGAME process is an interactive program used to accept and execute user orders; control platform motion, detections, and communications; determine engagement and other event outcomes; and display status information and tactical situations. The POST-GAME ANALYSIS process analyzes and lists critical data recorded during the exercise; supports exercise reconstruction; and tentatively evaluates some Measures of Effectiveness.

A global data area in NWISS, called the blackboard, is shared by all the major modules functioning during the exercise. It is the area where these modules interface with each other through the application of uniform naming conventions and the efficient use of memory. The blackboard is essentially comprised of numerous tables and subtables. Each table is assigned specific pointers while each subtable is assigned specific indices. The tables and subtables contain the fields (data) that are unique to that particular table or subtable. Each data item in the blackboard is referred to as a field and includes both whole words and specific bits. The field names are structured to provide the identity of the associated pointers and indices as well as the data type in the field.

Effective training at this level requires models of naval warfare interactions which provide realistic results based on an emulation of the actual warfare system. NWISS uses a wide variety of models to simulate the behavior of platforms, weapons, sensors, and communication systems. However, many of these models do not emulate the actual warfare system nor do they provide realistic results. Therefore, it is necessary to improve or replace these deficient models in order to obtain effective training and meet the objectives for which IBGTT was designed.

This thesis will address two models in particular. The first model is the Target Motion Analysis (TMA) Model which processes passive sonar contacts. The current model will be examined, followed by a presentation of a Kalman filter improvement to the model. Secondly, the Sonar Correlation Model will be examined, followed by a presentation of a Kalman filter to replace the current model.

The reader should be advised that the TMA Kalman Filter Model is in actuality equivalent to the Sonar Correlation Kalman Filter Model, as will become evident in the presentation of the two Kalman Filter Models. This thesis further presupposes that the reader is familiar with the Kalman filter.

II. SINGLE SENSOR TRACKING

A. TARGET MOTION ANALYSIS (TMA) MODEL

The current model will monitor the number of game minutes for which passive contact has been held on each target by each detecting passive sonar (i.e., submarine, surface sonar, towed array, or sonobuoy). When this time exceeds the TMA time defined by the user at simulation initiation a target motion analysis report will be displayed on the Passive Sonar Status Board. The TMA range, course, and speed displayed are derived as follows:

$$\text{TMA}\begin{bmatrix}\text{range}\\ \text{course}\\ \text{speed}\end{bmatrix} = \text{Actual target}\begin{bmatrix}\text{range}\\ \text{course}\\ \text{speed}\end{bmatrix} \pm \text{FACTOR}$$

The FACTOR is the product of a random number drawn from a uniform distribution and a derived parameter.

These parameters, which are indicated in Table I, cause increasingly accurate solutions to be developed as Signal Excess (SE) increases and as the target's true bearing changes (•B=delta B) from the true bearing of its initial detection. This latter factor simulates improved solutions derived from higher bearing rate targets and longer tracking times. The solution quality displayed will be selected from Table II as a function of SE and •B.

Once a TMA solution has been displayed, only the range is updated on the display. The range only update continues until the signal excess and/or change in true bearing cause a new parameter to be developed from the table (e.g., SE changes from -6 to -5 or •B changes from 5 to 6). When a new parameter is selected, a new TMA range, course, and quality

TABLE I

TMA Error Parameters

RELATIVE RANGE ERROR

SIGNAL EXCESS (SE) IN dB — BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12	R	.8R	.7R	.4R
-12 < SE ≤ -6	.8R	.7R	.4R	.25R
-6 < SE ≤ 0	.7R	.4R	.25R	.1R
0 < SE	.4R	.2R	.1R	.05R

COURSE ERROR IN DEGREES

SIGNAL EXCESS (SE) IN dB — BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12	120	90	45	30
-12 < SE ≤ -6	90	45	30	15
-6 < SE ≤ 0	45	30	15	7
0 < SE	30	15	7	5

RELATIVE SPEED ERROR

SIGNAL EXCESS (SE) IN dB — BEARING CHANGE IN DEGREES (•B)	•B≤5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12	.5S	.4S	.3S	.2S
-12 < SE ≤ -6	.4S	.3S	.2S	.1S
-6 ≤ SE ≤ 0	.3S	.2S	.1S	.05S

0 < SE	.2S	.1S	.05S	0

R = Actual Range
S = Actual Speed

TABLE II

TMA Quality

SIGNAL EXCESS (SE) IN dB / BEARING CHANGE IN DEGREES (•B)	•B<5	5<•B≤15	15<•B≤45	45<•B
SE ≤ -12	POOR	POOR	FAIR	FAIR
-12 ≤ SE ≤ -6	POOR	FAIR	FAIR	FAIR
-6 ≤ SE ≤ 0	FAIR	FAIR	FAIR	GOOD
0 < SE	FAIR	FAIR	GOOD	GOOD

will be calculated and displayed. Between TMA changes, the displayed range is updated each simulation cycle to show the range of the target estimated from the TMA course and speed.

If contact is lost for a time greater than the user-defined track loss time, the TMA solution will no longer be displayed. At any subsequent redetection of the same target, a new solution will be generated after the appropriate time interval has passed.

B. KALMAN FILTER IMPROVEMENTS

The model does not begin to compute a track until the TMA time defined by the user at simulation initiation is exceeded. This implies that all sensors and operators are equal, which is not realistic. Furthermore, this does not allow for accurate information to be used at time of detection unless the TMA time has already been exceeded. For instance, a passive sonobuoy dropped in front of a contact, producing a CPA (closest point of approach) for its initial detection, would have good track information that would not be utilized by the model; for only information after the TMA time is used in determining the FACTOR.

This TMA initiation time was included in the model because use of actual target information provided too accurate of a fix, even with bad sensor information, for a player to experience a realistic prosecution. The Kalman Filter Model eliminates the need for this waiting time since the model receives information as an operator would see it (that is, appparent bearing resulting from apparent position, including navigation error, and sensor bearing error). Thus, the Kalman Filter Model allows use of all sensor information with appropriate errors to provide realistic prosecution.

The TMA model attempts to simulate a changing area of probability (AOP) with improved solutions taken from the table as SE increases and true bearing changes. The problem of using true information instead of apparent has already been discussed. In addition, the improved AOP is heavily dependent upon the drawing of a random number. It has been observed in actual trainers that the AOP fluctuates as randomly as the random number generator, regardless of sensor information; providing confusing information to the player. Again, the Kalman Filter Model eliminates this

problem since the computed AOF updates smoothly with the sensor information.

C. KALMAN FILTER MODEL

First of all, it is assumed that during an encounter, the target's course and speed remain constant. The model updates the position of the fix since the last observation based on the previous estimate. This is based on the system model:

$$X(t) = PHI(t-1)*X(t-1)+W(t-1)$$

Thus, movement is:

State Extrapolation: $X\$hat(t) = PHI(t-1)*X\$hat(t-1)$

and

Error Covariance Extrapolation:

$$P(t) = PHI(t-1)*P(t-1)*PHI\$transpose(t-1) + Q(t-1)$$

where

X$hat is the estimated state vector. It is assumed to be multivariate normal with mean zero.

$$X(t) = [\ x(t) \quad y(t) \quad x' \quad y'\ ]\$transpose$$

x'= x velocity

y'= y velocity

PHI is the transition matrix. It describes how the state vector changes from X(t) to X(t+1).

$$PHI = \begin{bmatrix} 1 & 0 & \bullet t & 0 \\ 0 & 1 & 0 & \bullet t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \bullet t = \text{delta } t$$

P is the error covariance matrix.

P(t) = E[{X(t)-X$hat(t)}*{X(t)-X$hat(t)}$transpose]

W(t) is the plant noise. It describes the randomness of the system as it moves from state X(t) to X(t+1). W(t) is approximately N(0,Q(t)). For this model, Q is taken to be zero.

Next, a new fix is computed based on an observation. This is determined from the measurement model:

Z(t) = H(t)*X(t) + V(t)

Thus, measurement is:

Kalman Gain: K(t)=
P(t)*H$transpose(t)*[H(t)*P(t)*H$transpose(t) + R(t)]$^{-1}$,

State Update:
X$hat(t) := X$hat(t) + K(t)*[Z(t)-H(t)*X$hat(t)],

and

Error Covariance Update:
P(t):=P(t)-K(t)*[P(t)*H$transpose(t)]$transpose

where

:= indicates that the right hand side is computed and replaces the value on the left hand side of the symbol.

Z(t) is the actual measurement. The measurements are assumed to be linearly related to the system state X(t) by the observation matrix H(t). Note: H(t)*X$hat(t) is the predicted outcome of the measurement. The difference, Z(t)-H(t)*X$hat(t), is the measurement residual or shock.

V(t) is the measurement noise. It is approximately N(0,R(t)). For this model, all bearings are ± .5 degrees.

K(t) is the Kalman gain. The update from X$hat(t) just before the measurement to X$hat(t) just after the measurement is proportional to the shock; the Kalman gain is the proportionality constant.

It was stated earlier that the measurements are assumed to be linearly related to the system state. Since the measurement is in polar coordinates, h(x) is in fact nonlinear. Therefore, a transformation must be made on h(x) to give a linearly related H.

In this model, the observation will be made from a platform at (u,w) to a target at (x,y), where x is north and y is east. So,

$$h(X) = theta = \tan^{-1}[(y-w)/(x-u)]$$

or,

$$H = [\ \partial h(X)/\partial x \quad \partial h(X)/\partial y \quad \partial h(X)/\partial x' \quad \partial h(X)/\partial y'\ ]$$

@ represents the partial derivative

evaluated at X = X$hat. Hence,

$$H = [\ -\sin(theta)/range \quad \cos(theta)/range \quad 0 \quad 0\ ]$$

This model was built upon two initial conditions and two important assumptions. The initial conditions are:

E[X(0)] = X$hat(0)

and

E[{X(0)-X$hat(0)}*{X(0)-X$hat(0)}$transpose]= P(0)

where

X$hat(0) = [32cos$\theta_0$ 32sinθ_0 0 0]$transpose

θ_0 is the initial observation

and

$$P(0) = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 1000 \end{bmatrix}$$

Note: 1000 was chosen since it is approximately equal to ± one convergence zone (32 nm) and ± 32 knots.

The first assumption is that

E[W(k)*V(j)$transpose] = 0 for all j and k.

This means that the plant noise and the measurement noise are uncorrelated. Secondly, recall that the assumption is that during an encounter, the target's course and speed remain constant.

D. TRACK QUALITY

In order to reduce the number of changes required to the overall program, the TMA quality currently used from Table II will be utilized but based on different criteria. This will eliminate the need to change the blackboard; and more importantly, will not change the Status Tableau seen by the player, which is already full.

The track quality is based on the semi-major axis of the AOP. This is determined from the error covariance matrix as follows:

$$(\text{semi-major axis})^2 = (p11+p22)/2+\text{SQRT}[(p11-p22)^2/4+p12^2]$$

where pij are the elements of the P(t) matrix. A 2-sigma semi-major axis is used to insure a probability of 0.8647. The track quality is then determined as follows:

- If the 2-sigma axis is less than or equal to 500 yards, then the track receives a quality of GOOD. The criteria of 500 yards is used because the Engagement Model employs a 500 yard kill radius for a torpedo.
- If the 2-sigma axis is greater than 500 yards but less than or equal to 1000 yards, then the track receives a quality of FAIR. The criteria of 1000 yards was chosen simply because it is twice the GOOD criteria.
- If the 2-sigma axis is greater than 1000 yards, then the track receives a quality of POOR.

E. CHANGES TO OTHER SUBROUTINES

Implementation of this subroutine (LCLTMA, rational FORTRAN and source code listing are contained in Appendix A and B, respectively) will require some changes to other subroutines and additions to the blackboard. These changes and/or additions include:

1. Subroutine WARCYC

Each target needs a P(0) matrix in the blackboard; therefore, add:

```
      REAL PBB(4,4)
      REAL I_LCL$PMATRIX(4,4),I_RMT$PMATRIX(4,4)
      DO 50000 J=1,4
      DO 50001 K=1,4
      IF(.NOT.(J.EQ.K))GOTC 50001
      PBB(J,K) = 1000.
      GOTO 50000
50001 PBB(J,K) = 0.
50000 CONTINUE
      I_LCL$PMATRIX = PBB
      I_RMT$PMATRIX = PBB
```

2. Subroutine LCLPSN

- Change line 9 to read:

```
EQUIVALENCE (IBB,FBB,CBB,IBBW,IBBB,PBB)
```

This includes the blackboard of the P matrix.

- After line 113 add:

```
I_LCL$LASTTMATIME=(IAND(ISHFT(IBB(KPOINT_LCL
*+1),-0),65535)
```

This time is used in Subroutine LCLTMA.

- Change line 262 to read:

```
IF(.NOT.(I_LCL$OMNIFLAG.NE.1))GOTO 23269
```

This removes the TMA exceed time criteria.

- Change line 264 to read:

```
LCLTMA(KPOINT_LCL,RORLAT,RORLON,LBEAR,
*I_LCL$PMATRIX,I_LCL$LASTTMATIME)
```

III. MULTIPLE SENSOR TRACKING

A. SONAR CORRELATION MODEL

The Sonar Correlation Model would be more appropriately described as a procedure instead of a model. This routine determines the correlation of bearings between two detecting units at a specific target and performs two functions. First, the two detecting units (with an intersection angle of at least 60 degrees, or else the largest available angle) to a common target will display bearing lines. Second, multiple targets detected within a certain maximum arc will display only one line; and will set the composition field set (i.e., one, few, or many). All other passive sonar lines will not be displayed.

The Sonar Correlation Model can be turned on or off. This allows for the flexibility of being utilized for Battle Group Commanders and their staff when the "big picture" is the main concern and not the individual unit prosecution; and yet, be turned off when the trainer is being utilized for a single unit or group of units practicing coordinated operations.

B. KALMAN FILTER REPLACEMENT

The current procedure has many drawbacks. Only passive sonar lines are considered; active information is displayed separately and is not correlated with the passive information.

The routine searches through the Remote Table until it finds two bearing lines that meet the 60 degree criteria. These are displayed and the routine ceases to search; thus, not necessarily choosing the optimum solution. In addition,

no fix or track is associated with the correlation. As a result, each game minute two bearing lines are displayed (they may not be the same two from the previous minute nor necessarily an improvement) that jump around the screen, providing no useful information to the user.

The Kalman Filter Model permits all the information available, both active and passive, to be correlated on a specific target and be displayed as a fix with an updated track. In addition, the track quality associated with the fix provides the user with the added information about the relative size of the AOP. The on and off ability of the Sonar Correlation Model will be incorporated into the Kalman Filter Model for the previously stated reasons concerning its flexibility.

C. KALMAN FILTER MODEL

As mentioned in the introduction, the single sensor tracking model is in actuality equivalent to the multiple sensor tracking model. The difference being the scope of the information being processed. The single sensor model is a subset of the Passive Sonar Model; whereas, the Multiple sensor model is a separate entity correlating all available information. The basics of the models, including assumptions and initial conditions, are the same. Therefore, only the differences from the single sensor tracking model, presented in Chapter Two, will be discussed here.

The model will handle bearing and range measurements as well as bearing only measurements. The bearings are ±.5 degrees and the ranges are ±.5 nautical miles. For the bearing only measurement, the H matrix will be the same as for the single sensor tracking model. For the bearing and range measurement, the following H matrix will be utilized:

$$H = \begin{bmatrix} -\sin(\text{theta})/\text{range} & \cos(\text{theta})/\text{range} & 0 & 0 \\ \cos(\text{theta}) & \sin(\text{theta}) & 0 & 0 \end{bmatrix}$$

The track quality will be the same as that presented in Chapter Two.

D. CHANGES TO OTHER SUBROUTINES

The rational FORTRAN and source code listing for this model are contained in Appendix C and D, respectively. The changes presented in Chapter Two for Subroutine WARCYC are also applicable to this model. The only other change that will be necessary is that all active information needs to be added to the Remote Table.

IV. TEST RESULTS

The single sensor and multiple sensor models have been tested on a limited basis. Each subroutine has been tested independently to insure that the models perform as designed. However, the subroutines have not been tested for integration into the overall NWISS program or other subroutines.

Due to time constraints and computer availability, integration tests were not possible. The added blackboard space required has not been made but should not pose any problems. Inherent with any new subroutine is the unforeseeable affect it may have on unrelated subroutines. This aspect of testing has not been performed.

APPENDIX A

SINGLE SENSOR MODEL (RATIONAL FORTRAN)

```
Subroutine LCLTMA(LCL$Pointer,          #LCL Table Pointer
                  RORLAT,               #LAT of BRG-line ORIGIN
                  RORLON,               #LON of BRG-line ORIGIN
                  LBEAR,                #integer SONAR TGT BRG
                                        #      with Heading error
                  I_LCL$PMATRIX,        #Kalman P matrix
                  I_LCL$LASTTMATIME)    #Game minute of last fix update
##################################################################################
#
# Purpose:LCLTMA determines TMA solutions for a detector,using a Kalman
#         filter, and stores data (for the Passive Sonar ASTAB) in the
#         LCL Table.
#
# Called by: LCLPSN
#
# Calls: MUL4X4  RRB2LL
#
##################################################################################

BBcommon
```

```
real  XEST(4),X(4)                      #System Model State Vector
      PHI(4,4)                          #System Model Transition Matrix
      PHIT(4,4)                         #Transpose of Transition Matrix
      P(4,4),I_LCL$PMATRIX(4,4)         #Error Covariance Matrix
      H(4)                              #Measurement Model Observation Matrix
      K(4)                              #Kalman Gain
      Z                                 #Measurement Vector

lcl$TMALAT$F=xlcl$TMALAT$F              #Estimated LAT/LON
lcl$TMALON$F=xlcl$TMALON$F              #                   of TGT
lcl$TMACSE$F=xlcl$TMACSE$F              #Estimated COURSE and
lcl$TMASPD$F=xlcl$TMASPD$F              #          SPEED of TGT

#MOVEMENT:
                                        #State Extrapolation:
      XEST(1)=X                         #Distance in north-south direction
      ANGPI(DELLON)                     #Insure shortway around earth
      XEST(2)=Y                         #Distance in east-west direction
      XEST(3)=X$dot                     #Speed vector in north-south direction
      XEST(4)=Y$dot                     #Speed vector in east-west direction

    #Initialize PHI Matrix
      PHI(1,3)=PHI(2,4)=$delta$time        #Movement time in hrs since last
```

```
                                     #update

    #X$hat=PHI*X$hat            #State Extrapolation Vector

    #PHI$transpose
    #P=PHI*P*PHI$transpose      #Error Covariance Extrapolation

#MEASUREMENT:

    #Estimated bearing
    #Estimated range

                                #Measurement Observation Matrix
     H(1)=-SIN(THETA$hat)/RNG$hat     #H$transpose=H
     H(2)=COS(THETA$hat)/RNG$hat
     H(3)=H(4)=0.

    #P*H$transpose              =(P*H$transpose)$transpose
    #H*P*H$transpose
    #H*P*H$transpose+R          R=BRG measurement error=±.5 degrees
    #Kalman Gain
    #H*X$hat
     ZHX=Z-HX                   #Measurement Residual
    #K*(Z-H*X$hat)
```

```
    #X$hat=X$hat+K*(Z-H*X$hat)     State Update

    #K*(P*H$transpose)$transpose
    #P=P-K*(P*H$transpose)$transpose   #Error Covariance Update

#New estimated bearing
#New estimated range

#Compute LAT/LON of BRG/RNG from ORIGIN
      CALL RRB2LL(_                    #Get LAT/LON
                  F_LCL$TMALAT,        #ORIGIN LAT -> FIX LAT (input/output)
                  F_LCL$TMALON,        #ORIGIN LON -> FIX LON (input/output)
                  RNG,                 #Range from ORIGIN to TGT
                  THETA,               #Bearing from ORIGIN to TGT
                  0.0,                 #Pass zero
                  COSL)                #Cosine of LAT (input/output)

putLCL$TMALAT$f                #New FIX position
putLCL$TMALON$f

#New estimated course
#New estimated speed
```

```
putLCL$TMACSE$f              #New FIX course
putLCL$TMASPD$f              #    and speed

#Determine semi-major axis of area of probability
    #SIGMA$squared=(P11+P22)/2+SQRT{((P11-P22)+(P11-P22))/4+P12*P12}
    #2SIGMA=2*SIGMA/2025 yds

if(2SIGMA <= 500 yds)then
   TMA$Quality=2             #GCOD
else if(2SIGMA > 500 yds & <= 1000 yds)then
   TMA$Quality=1             #FAIR
else(2SIGMA > 1000 yds)
   TMA$Quality=0             #PCOR

return
end           #End lclTMA
#######################################################################
Subroutine MUL4X4(A,         #4X4 matrix (input)
                  B,         #4X4 matrix (input)
                  C)         #4X4 matrix (output)
#######################################################################
#
```

```
# Purpose: Multiplies two 4X4 matrices together.
#
# Called by: LCLTMA  CORSNR
#
#######################################################################

C=A*B

return
end              #End MUL4X4
```

APPENDIX B

SINGLE SENSOR MODEL (SOURCE CODE)

```
 SUBROUTINE LCLTMA(KPOINT_LCL,RORLAT,RORLON,LBEAR,I_LCL$PMATRIX,I_L
*CL$LASTTMATIME)
 IMPLICIT REAL*8 (A,C)
 INTEGER IBB(1025),IBBP(6,85)
 INTEGER*2 IBBW(2,1025)
 BYTE IBBB(4,1025)
 REAL*8 CBB(512)
 REAL FBB(1025),I_LCL$PMATRIX(4,4),P(4,4),KPHTT(4,4),H(4),PHT(4)
 REAL K(4),HPHT,HPHTR,HX,PROD(4),XEST(4),X(4),PHI(4,4),PHIT(4,4)
 REAL PBB(4,1025)
 EQUIVALENCE (IBB,FBB,CBB,IBBW,IBBB,PBB)
 EQUIVALENCE (IBB(513),IBBP)
 COMMON/BBOARD/IBB
 F_LCL$TMALAT=(IAND(ISHFT(IBB(KPOINT_LCL+8),-0),65535)*1.*.0001-3.2
*)
 F_LCL$TMALON=(IAND(ISHFT(IBB(KPOINT_LCL+8),-16),65535)*1.*.0001-3.
*2)
 F_LCL$TMACSE=(IAND(ISHFT(IBB(KPOINT_LCL+5),-0),511))
 F_LCL$TMASPD=(IAND(ISHFT(IBB(KPOINT_LCL+4),-16),4095))
 XEST(1)=F_LCL$TMALAT-RORLAT
 DELLON=F_LCL$TMALON-RORLON
```

```
      ANGPI(DELLON)
      COST=COS(F_LCL$TMALAT)
      COSL=COS(RORLAT)
      XEST(2)=.5*(COSI+COST)*DELLON
      XEST(3)=F_LCL$TMASPD*COS(F_LCL$TMACSE)
      XEST(4)=F_LCL$TMASPD*SIN(F_LCL$TMACSE)
      DO 50002 J=1,4
      DO 50002 K=1,4
      IF(.NOT.(J.EQ.K)) GOTO 50003
      PHI(J,K)=1.
      GOTO 50002
50003 PHI(J,K)=0.
50002 CONTINUE
      PHI(1,3)=(IBB(103)-I_LCL$IASTTMATIME)/60.
      PHI(2,4)=PHI(1,3)
      DO 50004 J=1,4
      X(J)=0.
      DO 50004 K=1,4
50004 X(J)=X(J)+PHI(J,K)*XEST(K)
      DO 50005 J=1,4
      DO 50005 K=1,4
50005 PHIT(J,K)=PHI(K,J)
      CALL MUL4X4(PHI,I_LCL$PMATRIX,P)
      CALL MUL4X4(P,PHIT,I_LCL$PMATRIX)
```

```
      THETA=FATAN2(X(2),X(1))
      THETA=INT(THETA*(180./3.141592654)+.5)
      RNG=SQRT(X(1)*X(1)+X(2)*X(2))
      H(1)=-SIN(THETA)/RNG
      H(2)=COS(THETA)/RNG
      H(3)=0.
      H(4)=0.
      DO 50006 J=1,4
      PHT(J)=0.
      DO 50006 K=1,4
50006 PHT(J)=PHT(J)+I_LCL$PMATRIX(J,K)*H(K)
      HPHT=0.
      DO 50007 J=1,4
50007 HPHT=HPHT+H(J)*PHT(J)
      HPHTR=HPHT+.25
      DO 50008 J=1,4
50008 K(J)=PHT(J)/HPHTR
      Z=FLOAT(LBEAR)
      HX=0.
      DO 50009 J=1,4
50009 HX=HX+H(J)*X(J)
      ZHX=Z-HX
```

```
      DO 50010 J=1,4
50010 PROD(J)=K(J)*ZHX
      DO 50011 J=1,4
50011 XEST(J)=X(J)+PROD(J)
      DO 50012 J=1,4
      DO 50012 L=1,4
50012 KPHTT(J,L)=K(J)*PHT(L)
      DO 50013 J=1,4
      DO 50013 K=1,4
50013 I_LCL$PMATRIX(J,K)=I_LCL$PMATRIX(J,K)-KPHTT(J,K)
      THETA=FATAN2(XEST(2),XEST(1))
      THETA=INT(THETA*(180./3.141592654)+.5)
      RNG=SQRT(XEST(1)*XEST(1)+XEST(2)*XEST(2))
      F_LCL$TMALAT=RORLAT
      F_LCL$TMALON=RORLON
      COSL=COS(F_LCL$TMALAT)
      CALL ERB2LL(F_LCL$TMALAT,F_LCL$TMALON,RNG,THETA,0.,COSL)
      IBB(KPOINT_LCL+8)=IOR(IAND(IBB(KPOINT_LCL+8),NOT(ISHFT(65535,0))),
     *ISHFT(IAND(INT(.5+(F_LCL$TMALAT--3.2)/.0001),65535),0))
      IBB(KPOINT_LCL+8)=IOR(IAND(IBB(KPOINT_LCL+8),NOT(ISHFT(65535,16)))
     *,ISHFT(IAND(INT(.5+(F_LCL$TMALON--3.2)/.0001),65535),16))
      CSE=FATAN2(XEST(4),XEST(3))
      CSE=INT(CSE*(180./3.141592654)+.5)
      SPD=SQRT(XEST(3)*XEST(3)+XEST(4)*XEST(4))
```

```
 IBB(KPOINT_LCL+5)=IOR(IAND(IBB(KPOINT_LCL+5),NOT(ISHFT(511,0))),IS
*HFT(IAND((CSE),511),0))
 IBB(KPOINT_LCL+4)=IOR(IAND(IBB(KPOINT_LCL+4),NOT(ISHFT(4095,16))),
*ISHFT(IAND((SPD),4095),16))
 CONST1=I_LCL$PMATRIX(1,1)-I_LCL$PMATRIX(2,2)
 CONST2=I_LCL$PMATRIX(1,2)*I_LCL$PMATRIX(1,2)
 CONST=SQRT(CONST1*CONST1/4.+CONST2)
 I_LCL$SIGMASQR=(I_LCL$PMATRIX(1,1)+I_LCL$PMATRIX(2,2))/2.+CONST
 I_LCL$2SIGMA=SQRT(I_LCL$SIGMASQR)*2./2025.
 IF(I_LCL$2SIGMA.LE.500)THEN
   I_LCL$TMAQUALITY=2
 ELSE IF(I_LCL$2SIGMA.GT.500.AND.I_LCL$2SIGMA.LE.1000)THEN
   I_LCL$TMAQUALITY=1
 ELSE
   I_LCL$TMAQUALITY=0
 END IF
 IBB(KPOINT_LCL+9)=IOR(IAND(IBB(KPOINT_LCL+9),NOT(ISHFT(3,4))),ISHF
*T(IAND((I_LCL$TMAQUALITY),3),4))
 FETURN
 END
 SUBROUTINE MUL4X4(A,B,C)
 DIMENSION A(4,4),B(4,4),C(4,4)
 DO 60000 I=1,4
 DO 60001 J=1,4
```

```
      S=0.
      DO 60002 K=1,4
60002 S=S+A(I,K)*B(K,J)
60001 C(I,J)=S
60000 CONTINUE
      RETURN
      END
```

APPENDIX C

MULTIPLE SENSOR MODEL (RATIONAL FORTRAN)

```
Subroutine CORSNR
#########################################################################
#
# Purpose: (1) Correlate all sonar contacts (active and passive) and
#              store the updated FIX (Posit,CSE,SPD).
#          (2) Determine a TMA quality based on the criteria:
#              If the semi-major axis of the area of probability is:
#                (a) <= 500 yds
#                    TMA quality is GOOD
#                (b) > 500 yds & <= 1000 yds
#                    TMA quality is FAIR
#                (c) > 1000 yds
#                    TMA quality is POOR
#
# Called by: WARCYC
#
# Calls: CORR_SORT  MUL4X4  RRB2LL
#
#########################################################################
```

```
BBcommon
CORR$ccmmon

real XEST(4),X(4)                 #System Model State Vector
     PHI(4,4)                     #System Model Transition Matrix
     PHIT(4,4)                    #Transpose of Transition Matrix
     P(4,4),I_LCL$PMATRIX(4,4)    #Error Covariance Matrix
     H(4)                         #Measurement Model Observation Matrix
     KG(4),KGAIN(4,2)             #Kalman Gain
     Z(2)                         #Measurement Vector
     R(2,2)                       #Measurement Noise

                                  #Loop thru Remote Table
for(RMT$Pointer$First; RMT$Pointer$Valid; RMT$Pointer$Next)
   {
   if(xRMT$InUse$I==$no)next       #Finf the right slots
   RMT$DetectionType$i=xRMT$DetectionType$i
   if(RMT$DetectionType$i=$Sonar$Code)      #If sonar, set composition
     putRMT$Compositicn$I(1)                #    to 1
   }

if(Correlate$Sonar==$no)return
```

```
#_______________________________________________________________________
#Loop for each BLUE/ORANGE View

for(iview=$firstBLUE$view; iview<=$lastORANGE$view; iview=iview+1)
   {
   VUE$Pointer$To iview            #Get to the right view

   RMT$Pointer$To xVUE$FirstRmtIndx$i      #Set first and last
   istart=RMT$Pointer                      #    RMT Index as
   RMT$Pointer$To xVUE$LastRmtIndx$i       #    limits for loop
   iend=RMT$Pointer

   kore=0                          #Initialize counter
                                   #Loop for each RMT slot in View
   for(RMT$Pointer=istart; RMT$Pointer<=iend; RMT$Pointer$next)
      {
      if(XRMT$InUse$i==$no)next    #Skip if not in use
      RMT$DetectionType$i=xRMT$DetectionType$i    #Get Detection Type

      if(Correlate$Sonar==$yes & RMT$DetectionType$i==$Sonar$Code)
      *continue
      else next
```

```
    if(kore>= $Max$Corr)break            #Make sure that there are enough
                                         #    slots for the array
    kore=kore+1                          #Add to array counter
    irmtp(kore)=RMT$Pointer              #Save RMT Pointer
    idtor(kore)=xRMT$Detector$I          #Save Detector
    idtee(kore)=xRMT$Detectee$I          #Save Detectee
    ilast(kore)=xRMT$LastDetTime$I       #Save time of detection update
    ibear(kore)=xRMT$Bearing$I           #Save the bearing
    irnge(kore)=xRMT$Range$I             #Save the range
    ipnt(kore)=kore                      #Initialize sort index
    }

if(kore==0)return                   #Quit if no tracks

CALL CORR_SORT                      #Sort arrays by Detectee/Last-Det-Time

for(k=1; k<kore; k=j)
    {
    k1=ipnt(k)

    KPCINT_RMT=irmtp(k1)            #Set pointer

    RMT$TMALAT$F=xRMT$TMALAT$F           #Get posit,CSE,SPD
    RMT$TMALON$F=xRMT$TMALON$F           #    of last TMA estimate
```

```
        RMT$TMACSE$F=xRMT$TMACSE$F
        RMT$TMASPD$F=xRMT$TMASPD$F

        POS1$LAT$F=xPOS1$LAT$F      #Get posit of Detector
        POS1$LON$F=xPOS1$LON$F
        POS1$COSLAT$F=xPOS1$COSLAT$F

#MOVEMENT:
                          #State Extrapolation
        XEST(1)=X                   #Distance in north-south direction
        ANGPI(DELLON)               #Insure shortway around earth
        XEST(2)=Y                   #Distance in east-west direction
        XEST(3)=X$dot               #Speed vector in north-south direction
        XEST(4)=Y$dot               #Speed vector in east-west direction

        #Initialize the PHI matrix
           PHI(1,3)=PHI(2,4)=$delta$t   #Movement time in hrs since
                                        #    last update

        #X$hat=PHI*X$hat            State Extrapolation Vector

        #PHI$transpose
        #P=PHI*P*PHI$transpose      Error Covariance Extrapolation
```

```
#MEASUREMENT:

      for(j=k; j<=kore; j=j+1)
         {
         j1=ipnt(j)

         if(idtee(k1) != idtee(j1))break

         KPOINT_RMT=irmtp(j1)     #Set pointer

         RMT$DetecticnType=xRMT$DetectionType    #Get Detection Type

         #Estimated bearing
         #Estimated range

         if(k=j)                  #ORIGIN platform
           POS2$LAT$F=xPOS2$LAT$F   #Get posit of next detector
           POS2$LON$F=xPOS2$LON$F
           POS2$COSLAT$F=xPOS2$COSLAT$F

         #Adjust contact bearing to ORIGIN
         #X=north-south distance from ORIGIN to DETECTOR
         #Y=east-west distance from ORIGIN to DETECTOR
```

```
#THETAK=bearing from ORIGIN to DETECTOR
#DK=distance from ORIGIN to DETECTOR

ibear(j1)=DK*SIN(OBS BRG - BRG from ORIGIN)  #Observed bearing
                                             #adjusted to
                                             #ORIGIN platform

if(RMT$DetectionType != $Passive$Sonar) check for $Active$Sonar

                         #Measurement Observation Matrix
  H(1)=-SIN(THETA$hat)/RNG$hat      #H=H$transpose
  H(2)=COS(THETA$hat)/RNG$hat
  H(3)=H(4)=0

  #P*H$transpose          =(P*H$transpose)$transpose
  #H*P*H$transpose
  #H*P*H$transpose+R      R=BRG measurement error=±.5 degrees
  #Kalman Gain

  #Z(1)=measured bearing
  #H*X$hat
  #ZHX=Z(bearing)-H*X$hat     Measurement Residual
  #K*(Z-H*X$hat)
  #X$hat=X$hat+K*(Z-H*X$hat)  State Update
```

```
    #K* (P*H$transpose) $transpose
    #P=P-K* (P*H$transpose) $transpose      Error Covariance Update

if (RMT$DetectionType != $Active$Sonar) break

    #irnge (j1) =observed range adjusted to ORIGIN platform

                       #Measurement Observation Matrix
    HH (2, 1) =COS (THETA$hat)
    HH (2, 2) =SIN (THETA$hat)
    HH (1, 1) =-SIN (THETA$hat) /RNG$hat
    HH (1, 2) =COS (THETA$hat) /RNG$hat
    HH (1, 3) =HH (1, 4) =HH (2, 3) =HH (2, 4) =0

    #H$transpose
    #P*H$transpose
    #H*P*H$transpose
    #H*P*H$transpose+R     R=BRG measurement error=±.5 degrees
                           # =RNG measurement error=±.5 naut. mi.
    # (H*P*H$transpose+R) $inverse
    #Kalman Gain

    #Z (1) =measured bearing
```

```
    #Z(2)=measured range
    #H*X$hat
    #Z-H*X$hat               Measurement Residual
    #K*(Z-H*X$hat)
    #X$hat=X$hat+K*(Z-H*X$hat)   State Update

    #(P*H$transpose)$transpose
    #K*(P*H$transpose)$transpose
    #P=P-K*(P*H$transpose)$transpose      Error Covariance Update
   }

#New estimated bearing
#New estimated range

#Compute LAT/LON of BRG/RNG from ORIGIN
   CALL RRB2LL(_           #Get LAT/LON
        F_RMT$TMALAT       #ORIGIN LAT->FIX LAT (input/output)
        F_RMT$TMALON       #ORIGIN LON->FIX LON (input/output)
        RNG                #Range from ORIGIN to TARGET
        THETA              #Bearing from ORIGIN to TARGET
        0.0                #Pass zero
        COSL               #Cosine of latitude (input/output)

putRMT$TMALAT$f            #New FIX position
```

```
    putRMT$TMALON$f

    #New estimated course
    #New estimated speed

    putRMT$TMACSE$f                  #New FIX course
    putRMT$TMASPD$f                  #    and speed

    #Determine semi-major axis of area of probability
      #SIGMA$squared=(P11+P22)/2+SQRT{((P11-P22)+(P11-P22))/4+P12*P12}
      #2SIGMA=2*SIGMA/2025 yds

    if(2SIGMA <= 500 yds)then
      TMA$Quality=2                  #GOOD
    else if(2SIGMA > 500 yds & <= 1000 yds)then
      TMA$Quality=1                  #FAIR
    else(2SIGMA > 1000 yds)
      TMA$Quality=0                  #POOR

    }k=j                             #Set pointer to next detectee

}
```

```
                                                            SNR02320
      return                                                SNR02330
      end             #End CORSNR                           SNR02340
```

APPENDIX D

MULTIPLE SENSOR MODEL (SOURCE CODE)

```
      SUBROUTINE CORSNR
      IMPLICIT REAL*8 (A,C)
      INTEGER IBB(1025)
      INTEGER*2 IBBW(2,1025),IDTOR(800),IDTEE(800),ILAST(800),IBEAR(800)
      INTEGER*2 IRNGE(800),IPNT(800),KORE
      INTEGER*4 IRMTP(800)
      BYTE IBBB(4,1025)
      REAL*8 CBB(512)
      REAL FBB(1025),PBB(4,1025),H(4),PHT(4),KG(4),HPHT,HPHTR,HX,PROD(4)
      REAL KGAIN(4,2),I_RMT$PMATRIX(4,4),KPHTT(4,4),KKPHTT(4,4),R(2,2)
      REAL HH(2,4),HHT(4,2),PPHT(4,2),HHPHT(2,2),SUM(2,2),ADJ(2,2)
      REAL INVSUM(2,2),PHI(4,4),PHIT(4,4),P(4,4),HHX(2),PPHTT(2,4),X(4)
      REAL XEST(4),Z(2),KZHX(4)
      EQUIVALENCE (IBB,FBB,CBB,IBBW,IBBB,PBB)
      EQUIVALENCE (IBB(513),IBBP)
      COMMON/BBOARD/IBB
      COMMON/SCRPAD/IRMTP,IDTOR,IDTEE,ILAST,IBEAR,IRNGE,IPNT,KORE
      DATA R/.25,0.,0.,.25/
      KPOINT_RMT=IBBP(1,56)
70000 IF(.NOT.((KPOINT_RMT.GE.IBBP(1,56).AND.KPOINT_RMT.LT.(IBBP(1,56)+I
     *BBP(2,56))))GOTO 70001
```

```
      IF(IBB(KPOINT_RMT+8).EQ.0)GOTO 70002                              COR00220
      IBB(KPOINT_RMT+10)=IOR(IAND(IBB(KPOINT_RMT+10),NOT(ISHFT(1,29))),ICOR00230
     *SHFT(IAND((0),1),29))                                             COR00240
      IBB(KPOINT_RMT+10)=IOR(IAND(IBB(KPOINT_RMT+10),NOT(ISHFT(1,28))),ICOR00250
     *SHFT(IAND(('00000001'X),1),28))                                   COR00260
      IBB(111)=1                                                        COR00270
      I_RMT$DETECTIONTYPE=(IAND(ISHFT(IBB(KPOINT_RMT+8),-29),3))        COR00280
      IF(.NOT.(I_RMT$DETECTIONTYPE.EQ.2.OR.I_RMT$DETECTIONTYPE.EQ.24))GOCOR00290
     *TO 70002                                                          COR00300
      IBB(KPOINT_RMT+10)=IOR(IAND(IBB(KPOINT_RMT+10),NOT(ISHFT(3,23))),ICOR00310
     *SHFT(IAND((1),3),23))                                             COR00320
      IBB(KPOINT_RMT+10)=IOR(IAND(IBB(KPOINT_RMT+10),NOT(ISHFT(1,28))),ICOR00330
     *SHFT(IAND(('00000001'X),1),28))                                   COR00340
      IBB(111)=1                                                        COR00350
70002 KPOINT_RMT=KPOINT_RMT+15                                          COR00360
      GOTO 70000                                                        COR00370
70001 IF(IBB(256).EQ.0)GOTO 70099                                       COR00380
      IVIEW=IBB(129)                                                    COR00390
70003 IF(.NOT.(IVIEW.LE.IBB(132)))GOTO 70099                            COR00400
      KPOINT_VUE=IBBP(1,06)-1540+1540*IVIEW                             COR00410
      KPOINT_RMT=IBBP(1,56)-15+15*(IAND(ISHFT(IBB(KPOINT_VUE+1),-0),1638COR00420
     *3))                                                               COR00430
      ISTART=KPOINT_RMT                                                 COR00440
      KPOINT_RMT=IBBP(1,56)-15+15*(IAND(ISHFT(IBB(KPOINT_VUE+1),-14),163COR00450
```

```
     *83))
      IEND=KPOINT_RMT
      KORE=0
      KPOINT_RMT=ISTART
70004 IF(.NOT.(KPOINT_RMT.LE.IEND))GOTO 70005
      IF(IBB(KPOINT_RMT+8).EQ.0)GOTO 70006
      I_RMT$DETECTIONTYPE=(IAND(ISHFT(IBB(KPOINT_RMT+8),-29),3))
      IF(IBB(256).EQ.1.AND.(I_RMT$DETECTIONTYPE.EQ.2.OR.I_RMT$DETECTIONT
     *YPE.EQ.24))GOTO 70007
      GOTO 70006
70007 IF(KORE.GE.800)GOTO 70005
      KORE=KORE+1
      IRMTP(KORE)=KPOINT_RMT
      IDTOR(KORE)=(IAND(ISHFT(IBB(KPOINT_RMT+8),-10),1023))
      IDTEE(KORE)=(IAND(ISHFT(IEB(KPOINT_RMT+7),-0),1023))
      ILAST(KORE)=(IAND(ISHFT(IEB(KPOINT_RMT+2),-0),65535))
      IBEAR(KORE)=(IAND(ISHFT(IEB(KPOINT_RMT+7),-10),511))
      IRNGE(KORE)=(IAND(ISHFT(IEB(KPOINT-RMT+7),-0),511))
      IPNT(KORE)=KORE
70006 KPOINT_RMT=KPOINT_RMT+15
      GOTO 70004
70005 IF(KORE.EQ.0)GOTO 70099
      CALL CORR_SORT
      K=1
```

```
      J=K                                                               COR00700
70009 IF(.NOT.(K.LT.KORE))GOTO 70010                                    COR00710
      K1=IPNT(K)                                                        COR00720
      KPOINT_RMT=IRMTP(K1)                                              COR00730
      F_RMT$TMALAT=(IAND(ISHFT(IBB(KPOINT_RMT+8),-0),65535)*1.*.0001-3.2COR00740
     *)                                                                 COR00750
      F_RMT$TMALON=(IAND(ISHFT(IBB(KPOINT_RMT+8),-16),65535)*1.*.0001-3.COR00760
     *2)                                                                COR00770
      F_RMT$TMACSE=(IAND(ISHFT(IBB(KPOINT_RMT+5),-0),511))              COR00780
      F_RMT$TMASPD=(IAND(ISHFT(IBB(KPOINT_RMT+4),-16),4095))            COR00790
      F_POS1$LAT=FBB(KPOIONT_RMT)                                       COR00800
      F_POS1$LON=FBB(KPOINT_RMT+1)                                      COR00810
      F_POS1$COSLAT=FBB(KPOINT_RMT+13)                                  COR00820
      XEST(1)=F_RMT$TMALAT-F_POS1$LAT                                   COR00830
      DELLON=F_RMT$TMALON-F_POS1$LON                                    COR00840
      ANGPI(DELLON)                                                     COR00850
      COSL=COS(F_RMT$TMALAT)                                            COR00860
      XEST(2)=.5*(COSL+F_POS1$COSLAT)*DELLON                            COR00870
      XEST(3)=F_RMT$TMASPD*COS(F_RMT$TMACSE)                            COR00880
      XEST(4)=F_RMT$TMASPD*SIN(F_RMT$TMACSE)                            COR00890
      DO 70011 JJ=1,4                                                   COR00900
      DO 70011 KK=1,4                                                   COR00910
      IF(.NOT.(JJ.EQ.KK))GOTO 70012                                     COR00920
      PHI(JJ,KK)=1.                                                     COR00930
```

```
      GOTO 70011
70012 PHI(JJ,KK)=0.
70011 CONTINUE
      F_RMT$DELTIME=(IBB(103)-IIAST(K1))/60.
      PHI(1,3)=F_RMT$DELTIME
      PHI(2,4)=F_RMT$DELTIME
      DO 70013 JJ=1,4
      X(JJ)=0.
      DO 70013 KK=1,4
70013 X(J)=X(J)+PHI(JJ,KK)*XEST(KK)
      DO 70014 JJ=1,4
      DO 70014 KK=1,4
70014 PHIT(JJ,KK)=PHT(KK,JJ)
      CALL MUL4X4(PHI,I_RMT$PMATRIX,P)
      CALL MUL4X4(P,PHIT,I_RMT$PMATRIX)
70015 IF(.NOT.(J.LE.KORE))GOTO 70016
      J1=IPNT(J)
      IF(IDTEE(K1).NE.IDTEE(J1))GOTO 70017
      KPOINT_RMT=IRMTP(J1)
      I_RMT$DETECTIONTYPE=(IAND(ISHFT(IBB(KPOINT_RMT+8),-29),3))
      THETA=FATAN2(X(2),X(1))
      THETA=INT(THETA*(180./3.141592654)+.5)
      RNG=SQRT(X(1)*X(1)+X(2)*X(2))
      IF(K.EQ.J)GOTO 70008
```

```
      F_POS2$LAT=FBB(KPOINT_RMT)                                        COR01180
      F_POS2$LON=FBB(KPOINT_RMT+1)                                      COR01190
      F_POS2$COSLAT=FBB(KPOINT_RMT+13)                                  COR01200
      X=F_POS2$LAT-F_POS1$LAT                                           COR01210
      Y=F_POS2$LON-F_POS1$LON                                           COR01220
      ANGPI(Y)                                                          COR01230
      COSL=COS(F_POS2$LAT)                                              COR01240
      Y=.5*(F_POS2$COSLAT+F_POS1$COSLAT)*Y                              COR01250
      THETAK=FATAN2(Y,X)                                                COR01260
      THETAK=INT(THETAK*(180./3.141592654)+.5)                          COR01270
      DK=SQRT(X*X+Y*Y)                                                  COR01280
      IBEAR(J1)=DK*SIN(IBEAR(J1)-THETAK)                                COR01290
70008 IF(.NOT.(I_RMT$DETECTIONTYPE.EQ.2))GOTO 70018                     COR01300
      H(1)=-SIN(THETA)/RNG                                              COR01310
      H(2)=COS(THETA)/RNG                                               COR01320
      H(3)=0.                                                           COR01330
      H(4)=0.                                                           COR01340
      DO 70019 JJ=1,4                                                   COR01350
      PHT(JJ)=0.                                                        COR01360
      DO 70019 KK=1,4                                                   COR01370
70019 PHT(JJ)=PHT(JJ)+I_RMT$PMATRIX(JJ,KK)*H(KK)                        COR01380
      HPHT=0.                                                           COR01390
```

```
      DO 70020 JJ=1,4                                                  COR01400
70020 HPHT=HPHT+H(JJ)*PHT(JJ)                                          COR01410
      HPHTR=HPHT+.25                                                   COR01420
      DO 70021 JJ=1,4                                                  COR01430
70021 KG(JJ)=PHT(JJ)/HPHTR                                             COR01440
      Z(1)=FLOAT(IBEAR(J1))                                            COR01450
      HX=0.                                                            COR01460
      DO 70022 JJ=1,4                                                  COR01470
70022 HX=HX+H(JJ)*X(JJ)                                                COR01480
      ZHX=Z(1)-HX                                                      COR01490
      DO 70023 JJ=1,4                                                  COR01500
70023 PROD(JJ)=KG(JJ)*ZHX                                              COR01510
      DO 70024 JJ=1,4                                                  COR01520
70024 X(JJ)=X(JJ)+PROD(JJ)                                             COR01530
      DO 70025 JJ=1,4                                                  COR01540
      DO 70025 KK=1,4                                                  COR01550
70025 KPHTT(JJ,KK)=KG(JJ)*PHT(KK)                                      COR01560
      DO 70026 JJ=1,4                                                  COR01570
      DO 70026 KK=1,4                                                  COR01580
70026 I_RMT$PMATRIX(JJ,KK)=I_RMT$PMATRIX(JJ,KK)-KPHTT(JJ,KK)           COR01590
      GOTO 70027                                                       COR01600
70018 IF(.NOT.(I_RMT$DETECTIONTYPE.EQ.24))GOTO 70027                   COR01610
      IF(K.EQ.J)GOTO 70028                                             COR01620
```

```
      IRNGE(J1)=SQRT(DK*DK+IRNGE(J1)*IRNGE(J1))                          COR01630
70028 HH(2,1)=COS(THETA)                                                 COR01640
      HH(2,2)=SIN(THETA)                                                 COR01650
      HH(1,1)=-H(2,2)/RNG                                                COR01660
      HH(1,2)=H(2,1)/RNG                                                 COR01670
      HH(1,3)=0.                                                         COR01680
      HH(1,4)=0.                                                         COR01690
      HH(2,3)=0.                                                         COR01700
      HH(2,4)=0.                                                         COR01710
      DO 70029 JJ=1,4                                                    COR01720
      DO 70029 KK=1,4                                                    COR01730
70029 HHT(JJ,KK)=HH(KK,JJ)                                               COR01740
      DO 70030 JJ=1,4                                                    COR01750
      DO 70031 KK=1,2                                                    COR01760
      S=0.                                                               COR01770
      DO 70032 LL=1,4                                                    COR01780
70032 S=S+I_RMT$PMATRIX(JJ,LL)*HHT(LL,KK)                                COR01790
70031 PPHT(JJ,KK)=S                                                      COR01800
70030 CONTINUE                                                           COR01810
      DO 70033 JJ=1,2                                                    COR01820
      DO 70034 KK=1,2                                                    COR01830
      S=0.                                                               COR01840
```

```
      DO 70035 LL=1,4                                                   COR01850
70035 S=S+HH(JJ,LL)*PPHT(LL,KK)                                         COR01860
70034 HHPHT(JJ,KK)=S                                                    COR01870
70033 CONTINUE                                                          COR01880
      DO 70036 JJ=1,2                                                   COR01890
      DO 70036 KK=1,2                                                   COR01900
70036 SUM(JJ,KK)=HHPHT(JJ,KK)+R(JJ,KK)                                  COR01910
      DET=SUM(1,1)*SUM(2,2)-SUM(1,2)*SUM(2,1)                           COR01920
      ADJ(1,1)=SUM(2,2)                                                 COR01930
      ADJ(1,2)=-SUM(1,2)                                                COR01940
      ADJ(2,1)=-SUM(2,1)                                                COR01950
      ADJ(2,2)=SUM(1,1)                                                 COR01960
      DO 70037 JJ=1,2                                                   COR01970
      DO 70037 KK=1,2                                                   COR01980
70037 INVSUM(JJ,KK)=ADJ(JJ,KK)/DET                                      COR01990
      DO 70038 JJ=1,4                                                   COR02000
      DO 70039 KK=1,2                                                   COR02010
      S=0.                                                              COR02020
      DO 70040 LL=1,2                                                   COR02030
70040 S=S+PPHT(JJ,LL)*INVSUM(LL,KK)                                     COR02040
70039 KGAIN(JJ,KK)=S                                                    COR02050
70038 CONTINUE                                                          COR02060
      Z(1)=FLOAT(IBEAR(J1))                                             COR02070
      Z(2)=FLOAT(IRNGE(J1))                                             COR02080
```

```
      DO 70041  JJ=1,2
      S=0.
      DO 70042  KK=1,4
70042 S=S+HH(JJ,KK)*X(KK)
70041 HHX(JJ)=S
      DO 70043  JJ=1,2
70043 HHX(JJ)=Z(JJ)-HHX(JJ)
      DO 70044  JJ=1,4
      S=0.
      DO 70045  KK=1,2
70045 S=S+KGAIN(JJ,KK)*HHX(KK)
70044 KZHX(JJ)=S
      DO 70046  JJ=1,4
70046 X(JJ)=X(JJ)+KZHX(JJ)
      DO 70047  JJ=1,2
      DO 70047  KK=1,4
70047 PPHTT(JJ,KK)=PPHT(KK,JJ)
      DO 70048  JJ=1,4
      DO 70049  KK=1,4
      S=0.
      DO 70050  LL=1,2
70050 S=S+KGAIN(JJ,LL)*PPHTT(LL,KK)
70049 KKPHTT(JJ,KK)=S
70048 CONTINUE
```

```
      DO 70051 JJ=1,4
      DO 70051 KK=1,4
70051 I_RMT$PMATRIX(JJ,KK)=I_RMT$PMATRIX(JJ,KK)-KKPHTT(JJ,KK)
70027 J=J+1
      GOTO 70015
70017 THETA=FATAN2(X(2),X(1))
      THETA=INT(THETA*(180./3.141592654)+.5)
      RNG=SQRT(X(1)*X(1)+X(2)*X(2))
      F_RMT$TMALAT=F_POS1$LAT
      F_RMT$TMALON=F_POS1$LON
      COSL=F_POS1$COSLAT
      CALL RRB2LL(F_RMT$TMALAT,F_RMT$TMALON,RNG,THETA,0.,COSL)
      KPCINT_RMT=IRMTI([illegible])
      IBB(KPOINT_RMT+8)=IOR(IAND(IBB(KPOINT_RMT+8),NOT(ISHFT(65535,0))),
     *ISHFT(IAND(INT(.5+(F_RMT$TMALAT--3.2)/.0001),65535),0))
      IBB(KPOINT_RMT+8)=IOR(IAND(IBB(KPOINT_RMT+8),NOT(ISHFT(65535,16)))
     *,ISHFT(IAND(INT(.5+(F_RMT$TMALON--3.2)/.0001),65535),16))
      CSE=FATAN2(X(4),X(3))
      CSE=INT(CSE*(180./3.141592654)+.5)
      SPD=SQRT(X(3)*X(3)+X(4)*X(4))
      IBB(KPOINT_RMT+5)=IOR(IAND(IBB(KPOINT_RMT+5),NOT(ISHFT(511,0))),IS
     *HFT(IAND((CSE),511),0))
      IBB(KPOINT_RMT+4)=IOR(IAND(IBB(KPOINT_RMT+4),NOT(ISHFT(4095,16))),
     *ISHFT(IAND((SPD),4095),16))
```

```
      CONST1=I_RMT$PMATRIX(1,1)-I_RMT$PMATRIX(2,2)
      CCNST2=I_RMT$PMATRIX(1,2)*I_RMT$PMATRIX(1,2)
      CONST=SQRT(CONST1*CONST1/4.+CONST2)
      I_RMT$SIGMASQR=(I_RMT$PMATRIX(1,1)+I_RMT$PMATRIX(2,2))/2.+CONST
      I_RMT$2SIGMA=SQRT(I_RMT$SIGMASQR)*2./2025.
      IF(I_RMT$2SIGMA.LE.500)THEN
         I_RMT$TMAQUALITY=2
      ELSE IF(I_RMT$2SIGMA.GT.500.AND.I_RMT$2SIGMA.LE.1000)THEN
         I_RMT$TMAQUALITY=1
      ELSE
         I_RMT$TMAQUALITY=0
      END IF
      IBB(KPOINT_RMT+9)=IOR(IAND(IBB(KPOINT_RMT+9),NOT(ISHFT(3,4))),ISHF
     *T(IAND((I_RMT$TMAQUALITY),3),4))
70016 K=J
      GOTO 70009
70010 IVIEW=IVIEW+1
      GOTO 70003
70099 RETURN
      END
```

BIBLIOGRAPHY

B-K Dynamics, Inc., Battle Group Training Computer Support Facility (BGTCSF) Passive Sonar T&E Plan and Results, prepared for Naval Ocean Systems Center, San Diego, California, September 1983.

B-K Dynamics, Inc., Battle Group Training Computer Support Facility (BGTCSF) Target Motion Analysis, prepared for Naval Ocean Systems Center, San Diego, California, June 1983.

Pacer Systems, Inc., Interim Battle Group Tactical Trainer (IBGTT) Instructor User's Guide, Volume I, prepared for Naval Ocean Systems Center, San Diego, California, 17 January 1983.

Shudde, Rex H., A Multiple Leg TMA Procedure with Programs for the Hewlett-Packard HP-41CV, the Hewlett-Packard HP-75C, the Sharp PC-1500 (TRS-80 PC-2), and the Radio Shack TRS-80 Model 100 Portable Computers, Naval Postgraduate School, Monterey, California, September 1983.

System Development Corp., NWISS Programmer/Analyst Guide, prepared for Naval Ocean Systems Center, San Diego, California, 31 May 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
3.	Commander Naval Ocean Systems Center ATTN: Code 41 San Diego, CA 92152	1
4.	LT. Keith N. Spangenberg Cruiser-Destroyer Group Five FPO, San Francisco 96601-4703	1

Thesis
S66636 Spangenberg
c.1 Multiple sensor tracking in the Interim Battle Group Tactical Trainer.

31 OCT 88 32447

Thesis
S66636 Spangenberg
c.1 Multiple sensor tracking in the Interim Battle Group Tactical Trainer.

thesS66636
Multiple sensor tracking in the Interim
3 2768 000 61319 4
DUDLEY KNOX LIBRARY